JOURNAL OF FUNDAMENTAL MATHEMATICS
AND APPLICATIONS (JFMA) VOL. 6 NO. 2 (NOV 2023)
Available online at www.jfma.math.fsm.undip.ac.id

EXPLORING PHYSICS-INFORMED NEURAL NETWORKS FOR SOLVING BOUNDARY LAYER PROBLEMS

Muchamad Harry Yudha Pratama[1], Agus Yodi Gunawan[2]*

[1] *Mathematics Master Study Program, Institut Teknologi Bandung*
[2] *Industrial and Financial Mathematics Research Group, Institut Teknologi Bandung*
Jln. Ganesa 10, Bandung, Indonesia
Email: [1]harryyudha7@gmail.com, [2]ayodi@itb.ac.id
*Corresponding author

Abstract. In this paper, we explore a cutting-edge technique called as Physics-Informed Neural Networks (PINN) to tackle boundary layer problems. We here examine four different cases of boundary layers of second-order ODE: a linear ODE with constant coefficients, a nonlinear ODE with homogeneous boundary conditions, an ODE with non-constant coefficients, and an ODE featuring multiple boundary layers. We adapt the line of PINN technique for handling those problems, and our results show that the accuracy of the resulted solutions depends on how we choose the most reliable and robust activation functions when designing the architecture of the PINN. Beside that, through our explorations we aim to improve our understanding on how the PINN technique works better for boundary layer problems. Especially, the use of the SiLU (Sigmoid-Weighted Linear Unit) activation function in PINN has proven to be particularly remarkable in handling our boundary layer problems.

Keywords: PINN, boundary layers, perturbations, activation functions.

I. Introduction

Dealing with thin boundary layers presents significant computational difficulties due to their inherently steep gradients. When analyzing momentum transport, these boundary layers are often turbulent, leading to high numerical costs in modeling. In heat and mass transport scenarios, thin boundary layers can arise even in the laminar regime due to reduced diffusivity. For instance, cardiovascular mass transport problems involve extremely thin concentration boundary layers caused by the very low diffusion coefficients of biochemicals in blood, making their numerical modeling exceptionally challenging [1].

Physics-Informed Neural Networks (PINN) have emerged as a promising paradigm in scientific computing, revolutionizing the way complex physical systems and differential equations are solved. By integrating domain-specific physics into the neural network architecture, PINN effectively leverage the laws of physics to guide their learning process. This unique capability makes them particularly well-suited for tackling diverse scientific and engineering problems, such as fluid dynamics, structural mechanics, and heat transfer, where accurate modeling and prediction are paramount. Despite their popularity, the robustness of PINN remains a concern in certain applications. The scientific machine learning community recognizes the need to develop robust and dependable models as a priority. Boundary layers present one of the critical challenges to the robustness of PINN.

In recent years, several modifications to the original Physics-Informed Neural Network (PINN) approach have been proposed to address specific limitations. To tackle the spectral bias in deep neural networks, which restricts the learning of high-frequency functions, Fourier feature networks have been integrated into PINN [2]. Other variants like Conservative PINN (cPINN) [3], extended PINN (XPINN) [4], and domain decomposition techniques [5] have emerged to leverage localized neural networks in areas with high gradients or complex patterns, enabling efficient learning of intricate functions. Some approaches have also explored enhanced local sampling of collocation or training points near regions with high gradients to improve convergence [6, 7]. However, despite these efforts, none of these techniques have thoroughly studied or effectively addressed the challenges posed by thin boundary layers. This study reveals that domain decomposition alone cannot resolve the issues caused by learning in thin boundary layers, given their highly localized and abrupt behavior. Furthermore, increasing the resolution of collocation points within the boundary layer does not lead to a resolution of PINN training issues. While PINN has been successfully applied to various advection-diffusion transport problems, including boundary layers, through optimal weighting of loss terms and a focus on low Peclet numbers [8, 9, 10, 11, 12, 13, 14], it has remained elusive in handling thin boundary layers with vanishing viscosity/diffusivity, presenting a significant challenge for PINN.

Besides, selecting the appropriate activation function in neural networks is of paramount importance as it directly influences the model's learning capacity and overall performance. Activation functions introduce non-linearity to the network, allowing it to learn complex patterns and relationships within the data. Choosing the right activation function can prevent issues like vanishing or exploding gradients, which can hinder convergence during training. Moreover, different activation functions are suited for specific tasks; for instance, the rectified linear unit (ReLU) is effective in tackling the vanishing gradient problem, while the sigmoid function is often used in binary classification tasks. By understanding the characteristics of activation functions and tailoring them to the network architecture and task at hand, we can ensure faster convergence, enhanced model accuracy, and improved generalization on unseen data, ultimately leading to more robust and reliable neural networks.

In this paper, we comprehensively explore the utilization of various activation functions within neural networks to determine which function that robust enough and compatible for addressing boundary layer problems. We employ these methodologies to analyze four distinct categories of boundary layer problems. Initially, we tackle a basic linear second-order ordinary differential equation (ODE) characterized by constant coefficients, serving as the simplest form of a boundary layer problem. Subsequently, we delve into a nonlinear ODE with homogeneous boundary conditions to demonstrate that PINN does not converge to trivial solutions. Furthermore, we examine ODEs with non-constant coefficients that give rise to interior boundary layers. Lastly, we address second-order ODEs involving multiple boundary layers.

The structure of the document is outlined as follows. Section II presents the explanation of perturbation theory to address the boundary layer problem, including the discussions of four chosen types of boundary layer problems. In Section III, we provide our results of utilizing PINN, and compare them with their provided analytical solutions. Concluding remarks are written in the last section.

II. Boundary Layer Problems

Let us consider a perturbation problem of the form

$$\varepsilon y'' + F(x, y, y'; \varepsilon) = 0, \quad 0 < \varepsilon \ll 1,$$

subject to appropriate boundary conditions. Note that, during our discussion we do not have a certain physical meaning for the variable y in above equation. However, for some cases the variable y can be interpreted, for instance, as the vertical displacement of an elastic string with fixed ends. Our equation is a singularly perturbed problem, which means that the solution found by the differential equation when $\varepsilon = 0$ behaves very differently from the solution when $\varepsilon \to 0$. This equation leading to what is known as a ”boundary layer”. There is a region where the solution rapidly changes within a small span, and the thickness of this region diminishes as ε tends to zero. To handle this, perturbation theory comes into play, where the solution is expressed using asymptotic expansions and is divided into inner and outer regions. A procedure to solve boundary layer problem using matched asymptotic expansion can be found in [15]. To be self-contained we here rewrite the procedure as follows:

1. Find and solve the differential equation in the outer region. The outer equation can be found by setting $\varepsilon = 0$. To solve this equation, assume that the solution can be expanded in powers of ε.

   $$y_{outer}(x) \sim y_{outer_0}(x) + \varepsilon y_{outer_1}(x) + \varepsilon^2 y_{outer_2}(x) + \dots.$$

   Then solve the equation for each order, $O(1), O(\varepsilon), \dots$.

2. Specify where the boundary layer occurs. Let the boundary layer be at x_l.

3. Define inner variable as

   $$\xi = \frac{x - x_l}{\delta(\varepsilon)},$$

   where $\delta(\epsilon)$ is a positive function that is yet to be determined, tending towards 0 as ϵ approaches zero. This function can be regarded as a scaling factor in the inner region.

4. Find a solution in the inner region by assuming that

   $$y(x; \varepsilon) = y_{inner}(\xi; \varepsilon).$$

   So, in this region the equation becomes

   $$\varepsilon \delta^{-2} \frac{\partial^2 y_{inner}}{\partial \xi^2} + F(x_l + \delta\xi, \delta^{-1} \frac{\partial y_{inner}}{\partial \xi}, y_{inner}; \varepsilon) = 0$$

   and then assume that the solution can be expanded in powers of ε.

   $$y_{inner}(\xi) \sim y_{inner_0}(\xi) + \varepsilon^{\alpha_1} y_{inner_1}(\xi) + \varepsilon^{\alpha_2} y_{inner_2}(\xi) + \dots$$

5. Based on step (4), carry out a balancing process between terms to find an appropriate $\delta(\varepsilon)$ (see [15]).

6. Apply a matching process between the inner and outer solution for every order. Briefly, the inner solution for $\xi \to \infty$ is enforced to match the outer solution for $x \to x_l$.

7. Solution for problem at hand is given by the following composite form :

$$y \sim y_{outer}(x) + y_{inner}(\xi) - \text{matching term}.$$

We attempt to address various cases of boundary layer problems using PINN. Conventional PINN approaches, including those with dense collocation points and domain decomposition, have proven ineffective in solving these singular perturbation problems. Arzani et al. [16] proposed a new approach employing PINN with separate neural networks to approximate solutions at different order in the outer and inner expansion. By applying matching conditions, a consistent solution was obtained. However, the paper did not specify which activation function is suitable and robust enough to handle differential equations with sharp gradients.

The matching condition requires that the output of the inner PINN solution as ξ approaches infinity aligns with the output of the corresponding outer PINN solution as x approaches x_l. However, directly taking the limit to infinity is problematic due to the limitation of computation. To address this challenge, inspired from the classical similarity solutions found in boundary layer theory [17], Arzani et al. [16] proposed a new variable, $z = \frac{\xi}{A}$, $0 < z < 1$, with a suitably large constant A. Hence, a value of limit $\xi \to \infty$ was approximated by $z \to 1$.

In our study, we apply PINN technique introduced by Arzani et al. [16] to four boundary layer problems of second-order ODE, by employing diverse activation functions. The cases considered are a linear ODEs with constant coefficients, a nonlinear ODEs with homogeneous boundary conditions, an ODEs with non-constant coefficients, and an ODEs featuring multiple boundary layers. The first two cases are taken from [16] as our benchmarks. In each of these cases, we consider ε as a small value appearing in the given equation, leading to the formation of boundary layers. We treat ε as the perturbation parameter.

2.1. Case 1: Equation with constant coefficient and non-homogeneous boundary conditions

The equation is given by

$$\varepsilon \frac{\partial^2 y}{\partial x^2} + (1+\varepsilon)\frac{\partial y}{\partial x} + y = 0, \tag{1}$$

with boundary conditions:

$$y(0) = 0, y(1) = 1.$$

The outer equation leads to

$$\frac{\partial y_{outer}}{\partial x} + y_{outer} = 0. \tag{2}$$

An asymptotic analysis of (1) for $\varepsilon \to 0$ reveals that the distinguished limit based on the dominant balance among terms leads to $\delta(\varepsilon) = \varepsilon$. Referring to the perturbation theory [18, 15], since the coefficient of $\frac{dy}{dx}$ of the original problem is greater than 0, the boundary layer will take place at $x_l = 0$. So, defining an inner variable $\xi = \frac{x}{\varepsilon}$, and substituting back into (1), we have the inner equation:

$$\frac{\partial^2 y_{inner}}{\partial \xi^2} + \frac{\partial y_{inner}}{\partial \xi} = 0.$$

Following the work of Arzani et al. [16], the inner equation is rescaled to $z = \frac{\xi}{A}$ to make the matching condition possible, Thus,

$$\frac{1}{A}\frac{\partial^2 y_{inner}}{\partial z^2} + \frac{\partial y_{inner}}{\partial z} = 0. \tag{3}$$

We choose $A = 10$ as in [16]. Then, the boundary and matching conditions are written as

$$y_{inner}(0) = 0,$$

$$y_{outer}(1) = 1,$$

$$\lim_{\xi \longrightarrow \infty} y_{inner} = \lim_{z \longrightarrow 1} y_{inner} = \lim_{x \longrightarrow 0^+} y_{outer.}$$

By combining the outer and inner solutions with the appropriate matching conditions at the boundary layer, we here approximate the solution up to the zeroth order of the perturbation terms. The inner solution accounts for the behavior near the boundary layer, while the outer solution handles the behavior away from the boundary layer. The value of A affects the accuracy of the approximation. For the present case, we have the following analytical solution:

$$y_{outer}(x) = e^{1-x},$$

$$y_{inner}(x) = e - e^{1-\frac{x}{\varepsilon}},$$

and in terms of composite solution for the leading order, we get

$$y(x) = e^{1-x} - e^{1-\frac{x}{\varepsilon}}. \tag{4}$$

We find (4) different from the solution from [16] because of the method used to find the analytical solution. Our methodology involves employing perturbation theory to distinguish between the inner and outer regions. In contrast, Arzani [16] utilized the characteristic polynomial approach.

2.2. Case 2: Equation with non-linear term and homogeneous boundary Conditions

The equation is provided by

$$\varepsilon\frac{\partial^2 y}{\partial x^2} + 2\frac{\partial y}{\partial x} + e^y = 0 \tag{5}$$

with the boundary conditions:

$$y(0) = y(1) = 0$$

This particular case holds considerable importance as it sheds light on an inherent limitation of PINN. Despite their powerful capabilities, there are instances where PINN struggle to overcome the complexities presented by certain functions, such as exponential functions, and homogeneous boundary conditions. Consequently, these challenges may lead to the emergence of trivial solutions. An equation for the leading order outer problem is given by

$$2\frac{\partial y_{outer}}{\partial x} + e^{y_{outer}} = 0. \tag{6}$$

Using a similar analysis as in Case 1, the boundary layer takes place at $x_l = 0.$. Furthermore, the distinguished limit leads to $\delta(\varepsilon) = \varepsilon$. So, the inner variable becomes: $\xi = \frac{x}{\varepsilon}$. Substituting back this variable into (5), for the leading order of inner equation we have

$$\frac{\partial^2 y_{inner}}{\partial \xi^2} + 2\frac{\partial y_{inner}}{\partial \xi} = 0.$$

In term of $z = \frac{\xi}{A}$, the inner equation now becomes:

$$\frac{1}{A}\frac{\partial^2 y_{inner}}{\partial z^2} + 2\frac{\partial y_{inner}}{\partial z} = 0. \tag{7}$$

The boundary and matching conditions that should be satisfied are

$$y_{inner}(0) = 0,$$

$$y_{outer}(1) = 0,$$

$$\lim_{\xi \longrightarrow \infty} y_{inner} = \lim_{x \longrightarrow 0^+} y_{outer},$$

The leading order analytical solutions to this problem are given by

$$y_{outer}(x) = \ln\frac{2}{x+1},$$

$$y_{inner}(x) = \ln(2)(1 - e^{-2\frac{x}{\varepsilon}}),$$

or in term of the composite solutions we obtain

$$y(x) = \ln\frac{2}{x+1} - \ln(2)e^{-2\frac{x}{\varepsilon}}.$$

2.3. Case 3: Equation with Non-constant Coefficient and Non-Homogeneous Boundary Condition

The equation is given by

$$\varepsilon\frac{\partial^2 y}{\partial x^2} + x\frac{\partial y}{\partial x} + xy = 0, \tag{8}$$

where the boundary conditions are

$$y(-1) = e/2, y(1) = 2/e.$$

The leading order for the outer equation is

$$x\frac{\partial y_{outer}}{\partial x} + xy_{outer} = 0. \tag{9}$$

Using a similar analysis as before, the boundary layer takes place at $x_l = 0$ which is an interior point, separating two outer regions, say $y^{(1)}_{outer}$ on the left and $y^{(2)}_{outer}$ on the right of the interior point. Via balancing process we obtain $\delta(\varepsilon) = \varepsilon^{\frac{1}{2}}$. So, we have the inner variable $\xi = \frac{x}{\varepsilon^{\frac{1}{2}}}$, and the inner equation becomes

$$\frac{\partial^2 y_{inner}}{\partial \xi^2} + \xi \frac{\partial y_{inner}}{\partial \xi} = 0.$$

Introducing the scale inner variable $z = \frac{\xi}{A}$, we obtain

$$\frac{1}{A^2}\frac{\partial^2 y_{inner}}{\partial z^2} + z\frac{\partial y_{inner}}{\partial z} = 0. \tag{10}$$

The boundary and the matching conditions are

$$y^{(1)}_{outer}(-1) = e/2,$$

$$y^{(2)}_{outer}(1) = 2/e,$$

$$\lim_{\xi \to -\infty} y_{inner} = \lim_{x \to x_l^-} y^{(1)}_{outer},$$

$$\lim_{\xi \to \infty} y_{inner} = \lim_{x \to x_l^+} y^{(2)}_{outer},$$

In this case, analytical solutions are not used to be compared with solutions from the neural network. Instead, the network's errors will be used to determine how accurate the solutions are.

2.4. Case 4: Multiple Boundary Layer

Let us consider

$$\varepsilon\frac{\partial^2 y}{\partial x} + (1-2x)\frac{\partial y}{\partial x} - 2y = 0, \tag{11}$$

with the boundary conditions

$$y(0) = -1, y(1) = 1.$$

The leading order for the outer equation is given by

$$(1-2x)\frac{\partial y_{outer}}{\partial x} - 2y_{outer} = 0. \tag{12}$$

Using a similar analysis as before, we identify the presence of two boundary layers, namely $x_l = 0$ and $x_l = 1$. After balancing process among terms, we obtain $\delta(\varepsilon) = \varepsilon$. Consequently, we introduce two corresponding inner variables: $\xi_1 = \frac{x}{\varepsilon}$ and $\xi_2 = \frac{x-1}{\varepsilon}$. Substituting back into equation (11), we obtain the inner equations

$$\begin{aligned} \frac{1}{A}\frac{\partial^2 y^{(1)}_{inner}}{\partial z^2} + \frac{\partial y^{(1)}_{inner}}{\partial z} &= 0, \\ \frac{1}{A}\frac{\partial^2 y^{(2)}_{inner}}{\partial z^2} - \frac{\partial y^{(2)}_{inner}}{\partial z} &= 0. \end{aligned} \tag{13}$$

The boundary and matching condition that must be satisfied are

$$y^{(1)}_{inner}(0) = -1,$$

$$y^{(2)}_{inner}(0) = 1,$$

$$\lim_{\xi \to \infty} y^{(1)}_{inner} = \lim_{x \to 0^+} y_{outer},$$

$$\lim_{\xi \to -\infty} y^{(2)}_{inner} = \lim_{x \to 1^-} y_{outer}.$$

Just like in Case 3, in this case, analytical solutions are not being used to compare with the neural network's solutions. Instead, the focus is on analyzing the network's errors to determine how accurate the solutions are.

III. Results

For our simulations, we employ distinct networks to train the solutions for both the inner and outer components, aiming to minimize the overall error. The error is defined as follows,

$$MSE = \sum_{i \in \{outer\}} MSE_i + \sum_{j \in \{inner\}} MSE_j + \sum_{k \in \{bmc\}} MSE_k$$

where MSE_i denotes the squares of the left-hand sides of the outer (inner, boundary and matching conditions). For example, for Case 1, the MSE is defined as follows:

$$\begin{aligned} MSE = (\frac{\partial \hat{y}_{outer}}{\partial x} + \hat{y}_{outer})^2 + (\frac{1}{A}\frac{\partial^2 \hat{y}_{inner}}{\partial z^2} + \frac{\partial \hat{y}_{inner}}{\partial z})^2 + \\ (\hat{y}_{inner}(0))^2 + (\hat{y}_{outer}(1) - 1)^2 + (\hat{y}_{inner}(z=1) - \hat{y}_{outer}(0))^2 \end{aligned} \tag{14}$$

where the $\hat{y}_{outer}, \hat{y}_{inner}$ are the network output for the outer and the inner solution, respectively.

Algorithm 1: Network Training Procedure

```
   Data: Outer differential equation, inner differential equation, boundary, and matching
         condition
   Result: Networks for the outer and inner regions
1  Initialize the number of hidden layers, dimension of hidden layers, batch size, and
     number of epochs (N);
2  Set the flag is_outer to false;
3  for $i = 1$ to $N$ do
4      if is_outer is false then
5          Solve the outer equation;
6          if outer solution is found then
7              is_outer = true;
8      else
9          Solve the inner equation and adjust the outer solution based on boundary and
             matching conditions;
10 return Networks for the outer and inner regions;
```

The pseudo-code detailing the procedure for training the networks is presented in Algorithm 1. Results for four cases are presented in the next subsections. Note that simulations apply to the leading-order term only.

3.1. Case 1

We construct neural network architecture consisting of five hidden layers, each containing 50 neurons. During the training process, the network is subjected to 60 distinct random collocation points within the domain for every iteration. Those number of hidden layers are just optional. However, for Case 4 (see detail discussion below) more number of neurons are needed to attain good error of calculations. Various activation functions are chosen to demonstrate the method, such as CELU [19], ELU [20], GELU [21], LeakyReLU [22], Mish [23], ReLU [24], SELU [25], SiLU [26], Sigmoid, Softmax [27], Softmin, Softplus [28], Tanh. The results for various activation functions are presented in Table 1.

Table 1. Network Errors in Solving Problem Case 1 with Different Activation Functions.

Activation Function	**Error**
GELU()	0.0012683691456913948
SiLU()	0.008932067081332207
Tanh()	0.010858217254281044
Mish()	0.011430460028350353
SELU()	0.02776332013309002
Softmax()	2.969837498326403
Softmin()	2.977969169616699
ELU(alpha=1.0)	4.531148910522461
CELU(alpha=1.0)	4.531148910522461
Softplus(beta=1, threshold=20)	5.172677040100098
Sigmoid()	5.269680023193359
ReLU()	35.36870574951172
LeakyReLU(negative_slope=0.01)	36.54730987548828

Among the activation functions tested, GELU and SiLU stand out as the most promising options, yielding significantly lower errors compared to others. These two activations enable the accurate approximation of both inner and outer solutions. In contrast, the other activation functions, such as ReLU(), LeakyReLU(negative_slope=0.01), and Softplus(beta=1, threshold=20), show substantially higher errors, indicating their limited suitability for effectively tackling this specific type of singular perturbation problem.

SiLU and GELU are fundamentally similar-shape activation functions. SiLU, formulated as

$$\text{SiLU}(x) = x\frac{1}{1+e^{-x}},$$

represents the multiplication of the input by the sigmoid function. Initially developed for reinforcement learning tasks [26], SiLU has also demonstrated remarkable performance in solving singular differential equations in this case. On the other hand, GELU was introduced by Hendrycks (2016) [21] defined as follows:

$$\text{GELU}(x) = \frac{x}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

and approximated with

$$\text{GELU}(x) = 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right)$$

The progression of the loss and the resulting solution obtained by the PINN are presented in Figure 1. Notably, the PINN solution has demonstrated a high level of accuracy in approximating both the inner and outer solutions. We also observe that the PINN boundary layer solution (red) is close to the PINN inner solution (blue) for $x \to 0$ and to the PINN outer solution (green) for $x \to 1$.

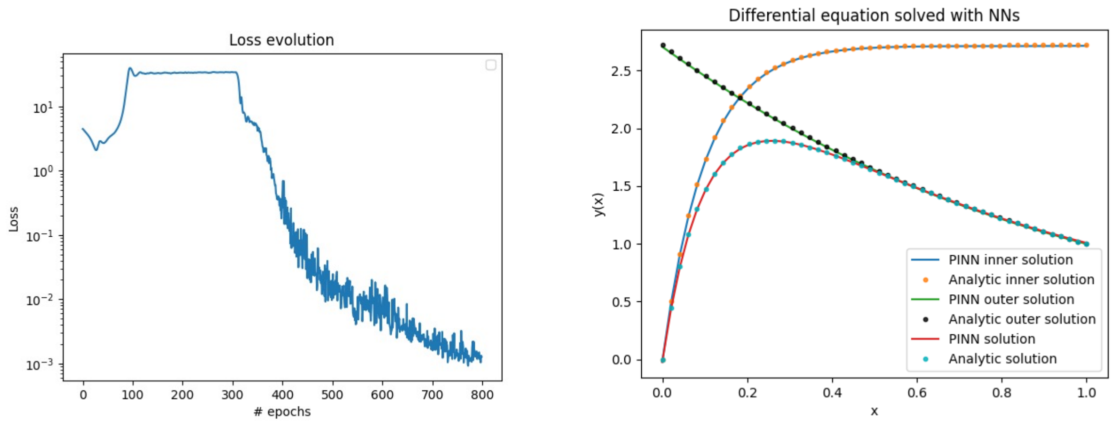

Figure 1. Evolution of Loss for the Network with Optimal Activation Function and the PINN Approximate Solution for Case 1

3.2. Case 2

Use the same architecture of previous case, we have results given in Table 2.

Table 2. Network Errors in Solving Problem Case 2 with Different Activation Functions.

Activation Function	Error
GELU(approximate='none')	0.0031172852031886578
Mish()	0.0037429218646138906
SiLU()	0.0054158661514520645
Softplus(beta=1, threshold=20)	0.7815308570861816
Tanh()	0.8075517416000366
ELU(alpha=1.0)	0.8107149600982666

CELU(alpha=1.0)	0.8107149600982666
SELU()	0.8469527959823608
Softmin(dim=None)	0.9993693232536316
Softmax(dim=None)	1.0017437934875488
ReLU()	2.0091211795806885
LeakyReLU(negative_slope=0.01)	3.824784278869629
Sigmoid()	4.371834754943848

We see two activation functions that were outstanding before, still did a good job for this problem. In addition, the Mish activation function can solve the problem in this case as well. The Mish activation function is defined as [23]:

$$\text{Mish}(x) = x \cdot \tanh(\ln(1 + e^x))$$

The progression of the loss and a comparison between the PINN solution and the analytic solution are illustrated in Figure 2. We can see that the PINN solution can approximate the analytical solution accurately. As same as in Case 1, we also observe that the PINN boundary layer solution (red) is close to the PINN inner solution (blue) for $x \to 0$ and to the PINN outer solution (green) for $x \to 1$.

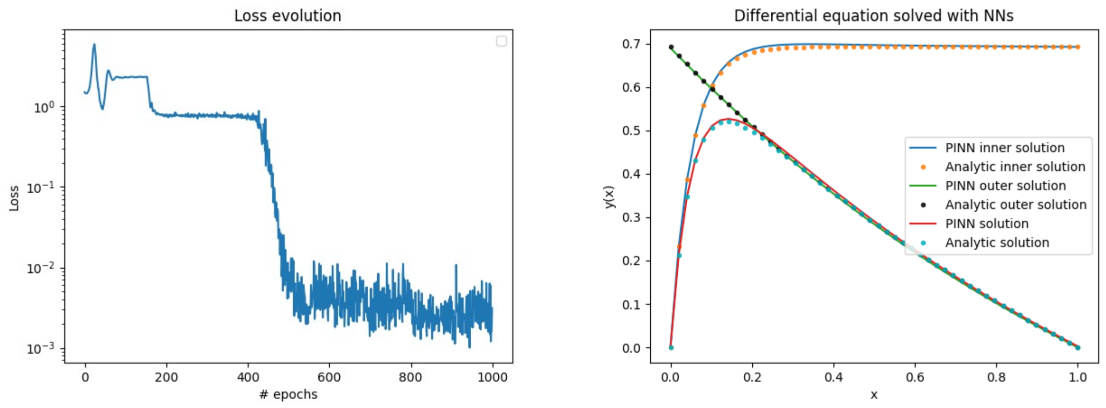

Figure 2. Evolution of Loss for the Network with Optimal Activation Function and the PINN Approximate Solution for Case 2

3.3. Case 3

For this interior point boundary layer problem, using the same network, we have results shown in Table 3.

Table 3. Network Errors in Solving Problem Case 3 with Different Activation Functions.

Activation Function	Error
Mish()	0.0012397984974086285
GELU(approximate='none')	0.001125918235629797
Tanh()	0.0020164770539849997
SiLU()	0.006233099848031998
CELU(alpha=1.0)	0.011121336370706558
ELU(alpha=1.0)	0.011121336370706558
Softplus(beta=1, threshold=20)	0.07702594995498657
ReLU()	0.09418447315692902
SELU()	0.040741708129644394
Softmin(dim=None)	1.900456428527832
Softmax(dim=None)	1.9024630784988403
Sigmoid()	17.83125114440918
LeakyReLU(negative_slope=0.01)	21.132722854614258

Interestingly, despite the increased complexity of this problem due to the presence of a boundary layer in the interior domain, several activation functions have proven to be effective in solving it. Among these options, the Mish activation function stands out as the most suitable choice. However, it is worth noting that the commonly used tanh activation function also performs well, yielding only minor errors in this context. Moreover, the SiLU and GELU activation functions demonstrate their ability to successfully tackle this case without encountering any significant challenges. The loss evolution and the PINN approximate solution are shown in Figure 3.. We also observe that the PINN boundary layer solution (red) is close to the PINN inner solution (blue) for x close to zero from both directions and to the PINN outer solution (green) for $x \to \pm 1$.

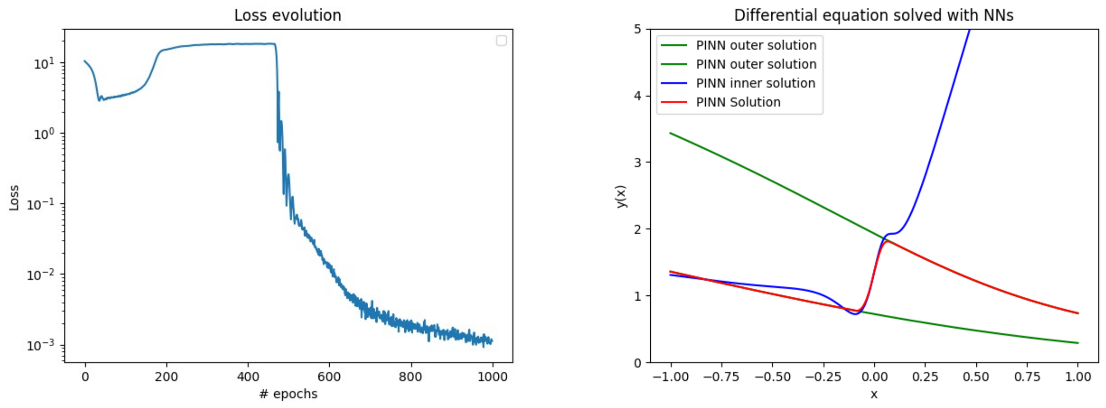

Figure 3. Evolution of Loss for the Network with Optimal Activation Function and the PINN Approximate Solution for Case 3

3.4. Case 4

In this scenario, employing only 50 neurons for each layer proved insufficient to achieve networks with errors close to zero. Consequently, a more intricate architecture with 80 neurons for each layer was constructed. Results for various activation functions are presented in Table 4.

Table 4. Network Errors in Solving Problem Case 4 with Different Activation Functions.

Activation Function	Error
SiLU()	0.0028864527121186256
Softplus(beta=1, threshold=20)	0.002719464246183634
Tanh()	0.5100778937339783
SELU()	3.809145212173462
ELU(alpha=1.0)	4.441868782043457
CELU(alpha=1.0)	4.441868782043457
LeakyReLU(negative_slope=0.01)	5.346591949462891
Softmax(dim=None)	9.852494239807129
Softmin(dim=None)	9.856649398803711
Sigmoid()	10.597221374511719
ReLU()	16.048564910888672
GELU(approximate='none')	16.8748836517334
Mish()	Failed

Remarkably, the previously effective GELU activation function exhibited complete failure in addressing this particular case. However, the SiLU activation function, on the other hand, continued to show its prowess in conquering this challenging problem. Despite the increased complexity of the situation, SiLU managed to deliver promising results and successfully tackle the task at hand. In Figure 4., the loss evolution and the approximate solution by modified-PINN are shown. We also observe that the PINN boundary layer solution (red) is close to the PINN inner solution (blue) for x close to 0 and to 1 (since both ends act as boundary layer points) and to the PINN outer solution (green) for $x \in (0, 1)$.

IV. Conclusions

We here demonstrated that Physics-Informed Neural Networks (PINN) offer a promising alternative for tackling boundary layer problems. The use of the SiLU activation function has proven to be particularly remarkable in handling differential equations, even those with sharp gradients. Its effectiveness in approximating solutions to complex equations and its ability to handle perturbations made it a valuable tool in the realm of scientific computing. PINN, along with advanced activation functions like SiLU, led to a great potential in efficiently solving a wide range of differential equations, providing researchers and engineers with valuable insights into various physical phenomena.

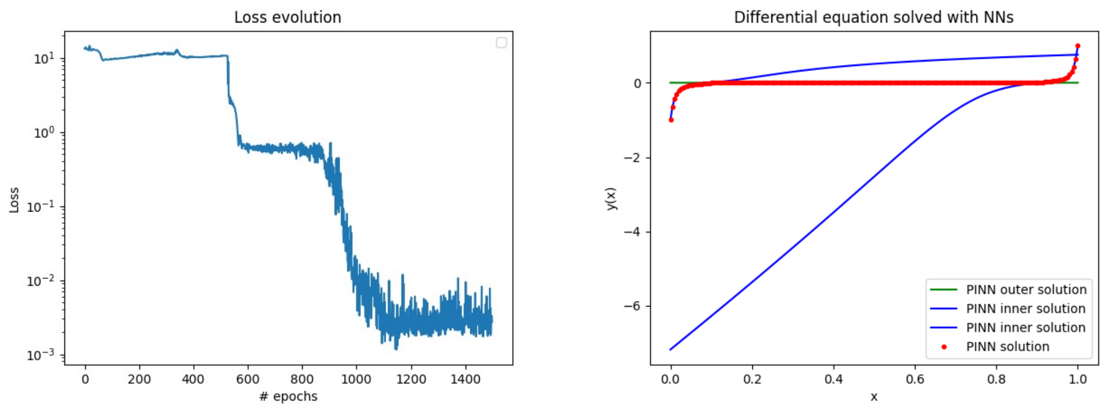

Figure 4. Evolution of Loss for the Network with Optimal Activation Function and the PINN Approximate Solution for Case 4

REFERENCES

[1] K. B. Hansen and S. C. Shadden, "A reduced-dimensional model for near-wall transport in cardiovascular flows," *Biomechanics and Modeling in Mechanobiology*, vol. 15, no. 3, pp. 713–722, 2016.

[2] S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 384, p. 113938, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782521002759

[3] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782520302127

[4] A. D. Jagtap and G. Em Karniadakis, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 28, no. 5, pp. 2002–2041, 2020. [Online]. Available: http://global-sci.org/intro/article$_detail/cicp/18403.html$

[5] H. Wang, R. Planas, A. Chandramowlishwaran, and R. Bostanabad, "Train once and use forever: Solving boundary value problems in unseen domains with pre-trained deep learning models," *ArXiv*, vol. abs/2104.10873, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:233346944

[6] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782519306814

[7] M. A. Nabian, R. J. Gladstone, and H. Meidani, “Efficient training of physics-informed neural networks via importance sampling,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 8, pp. 962–977, 2021.

[8] V. Dwivedi and B. Srinivasan, “Physics informed extreme learning machine (pielm)–a rapid method for the numerical solution of partial differential equations,” *Neurocomputing*, vol. 391, pp. 96–118, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219318144

[9] Q. He and A. M. Tartakovsky, “Physics-informed neural network method for forward and backward advection-dispersion equations,” *Water Resources Research*, vol. 57, no. 7, p. e2020WR029479, 2021.

[10] T. de Wolff, H. Carrillo, L. Martí, and N. Sanchez-Pi, “Towards optimally weighted physics-informed neural networks in ocean modelling,” *arXiv preprint arXiv:2106.08747*, 2021.

[11] R. Mojgani, M. Balajewicz, and P. Hassanzadeh, “Lagrangian pinns: A causality-conforming solution to failure modes of physics-informed neural networks,” *arXiv preprint arXiv:2205.02902*, 2022.

[12] A. Arzani, J.-X. Wang, and R. M. D’Souza, “Uncovering near-wall blood flow from sparse data with physics-informed neural networks,” *Physics of Fluids*, vol. 33, no. 7, 2021.

[13] L. Yang, X. Meng, and G. E. Karniadakis, “B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data,” *Journal of Computational Physics*, vol. 425, p. 109913, 2021.

[14] H. Bararnia and M. Esmaeilpour, “On the application of physics informed neural networks (pinn) to solve boundary layer thermal-fluid problems,” *International Communications in Heat and Mass Transfer*, vol. 132, p. 105890, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0735193322000124

[15] M. Holmes, *Introduction to Perturbation Methods*. Springer, 1995.

[16] A. Arzani, K. W. Cassel, and R. M. D’Souza, “Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation,” *Journal of Computational Physics*, vol. 473, p. 111768, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999122008312

[17] F. M. White, *Viscous fluid flow*. McGraw-Hill New York, 2006, vol. 3.

[18] A. Nayfeh, *Introduction to Perturbation Techniques*. John Wiley Sons, 1981.

[19] J. T. Barron, “Continuously differentiable exponential linear units,” *arXiv preprint arXiv:1704.07483*, 2017.

[20] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (ELUs),” in *International Conference on Learning Representations*, 2016.

[21] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[22] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, vol. 30, 2013, p. 3.

[23] D. Misra, "Mish: A self regularized non-monotonic activation function," 2020.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[25] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 971–980.

[26] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," 2017.

[27] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[28] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, "Improving deep neural networks using softplus units," in *International Joint Conference on Neural Networks*, 2015, pp. 1–4.